Tahiya Azad

Dr. Ankur Agrawal

CMPT 363

Assignment #8

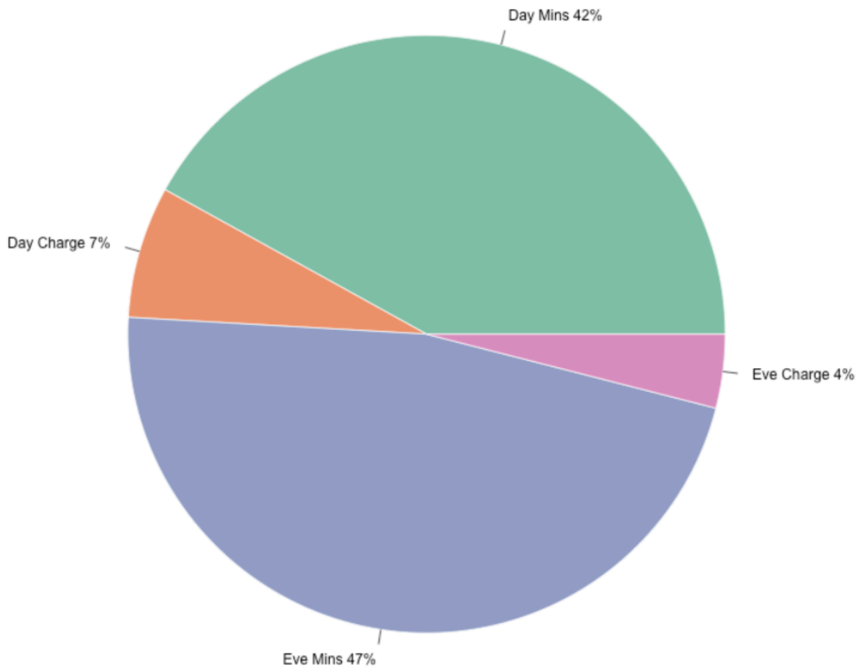
02 May 2024

The following results are 10 findings:

1) Pie Chart

Code:

```
options(repos = c(CRAN = "http://cran.rstudio.com"))
install.packages("dplyr")
install.packages("ggplot2")
library(dplyr)
library(ggplot2)
churn_data <- read.csv("/Users/tahiyaazad/Desktop/CMPT
436/DataMiningProject_2024/DM_2024/churn.txt")
churn_data <- as_tibble(churn_data)
options(width = 140)
print(churn_data)
library(RColorBrewer)

# Calculating total sums for Day Mins, Day Charge, Eve Mins, Eve Charge
Prop <- c(sum(churn_data$`Day.Mins`), sum(churn_data$`Day.Charge`),
sum(churn_data$`Eve.Mins`), sum(churn_data$`Eve.Charge`))
lbls <- c("Day Mins", "Day Charge", "Eve Mins", "Eve Charge")
pct <- round(Prop / sum(Prop) * 100)
lbls2 <- paste(lbls, " ", pct, "%", sep = "")
myPalatte <- brewer.pal(4, "Set2")
pie(Prop, labels = lbls2, border = "white", col = myPalatte, main = "The percentage of
Calls and Charges based on Day and Evening")
```



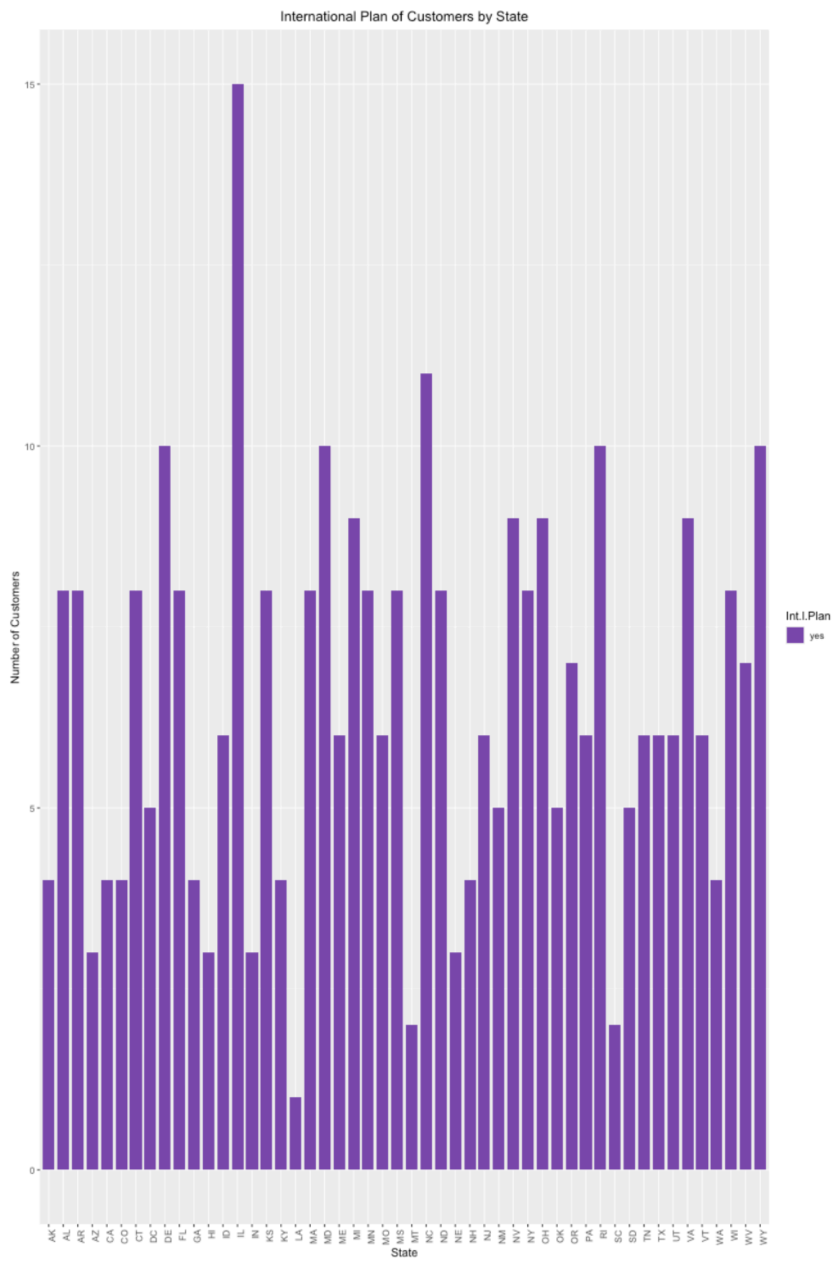
This pie chart shows the percentage of the calls and charges based on customers calls during the day and evening. If we look at the chart carefully, we will see that the percentage of day calls minutes is less than the percentage of evening calls minutes. But the charge for day calls is 7% which is greater than the Evening charge that is only 4%.

2) Bar Plot

Code:

```
options(repos = c(CRAN = "http://cran.rstudio.com"))
install.packages("dplyr")
install.packages("ggplot2")
library(dplyr)
library(ggplot2)
churn_data <- read.csv("/Users/tahiyaazad/Desktop/CMPT
436/DataMiningProject_2024/DM_2024/churn.txt")
churn_data <- as_tibble(churn_data)
options(width = 140)

int_l_plan_yes <- churn_data %>%
```

```
filter(Int.l.Plan == "yes") %>%
group_by(State, Int.l.Plan) %>%
summarise(Count = n(), .groups = "drop")

# Creating the bar plot
plot <- ggplot(int_l_plan_yes, aes(x = State, y = Count, fill = Int.l.Plan)) +
geom_bar(stat = "identity", width = 0.8) +
scale_fill_manual(values = c("yes" = "#8346b4")) +
theme(axis.text.x = element_text(angle = 90, hjust = 1), plot.title =
element_text(hjust = 0.5)) +
labs(x = "State", y = "Number of Customers", title = "International Plan of Customers
by State")

print(plot)
```

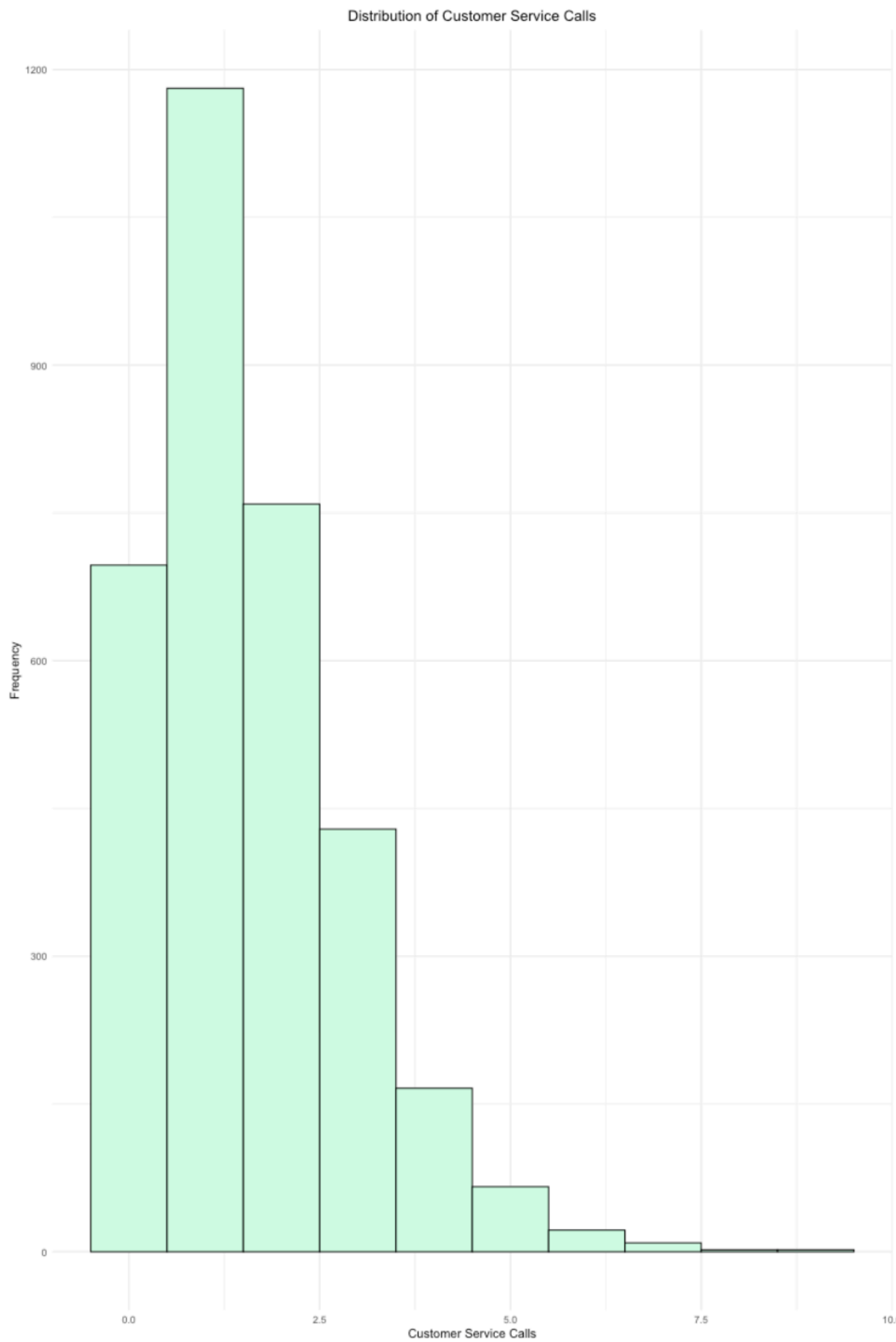


The bar plot illustrates the international plan of customers by different states. According to the plot, The state of IL has the largest number of customers that have international plan and LA has the lowest number of customers for international plan.

3) Histogram

Code:

```
int_l_plan_yes <- churn_data %>%
filter(Int.l.Plan == "yes") %>%
group_by(State, Int.l.Plan) %>%
summarise(Count = n(), .groups = "drop")

#Creating the bar plot
plot <- ggplot(int_l_plan_yes, aes(x = State, y = Count, fill = Int.l.Plan)) +
geom_bar(stat = "identity", width = 0.8) +
scale_fill_manual(values = c("yes" = "#8346b4")) +
theme(axis.text.x = element_text(angle = 90, hjust = 1), plot.title =
element_text(hjust = 0.5)) +
labs(x = "State", y = "Number of Customers", title = "International Plan of Customers
by State")

print(plot)
```

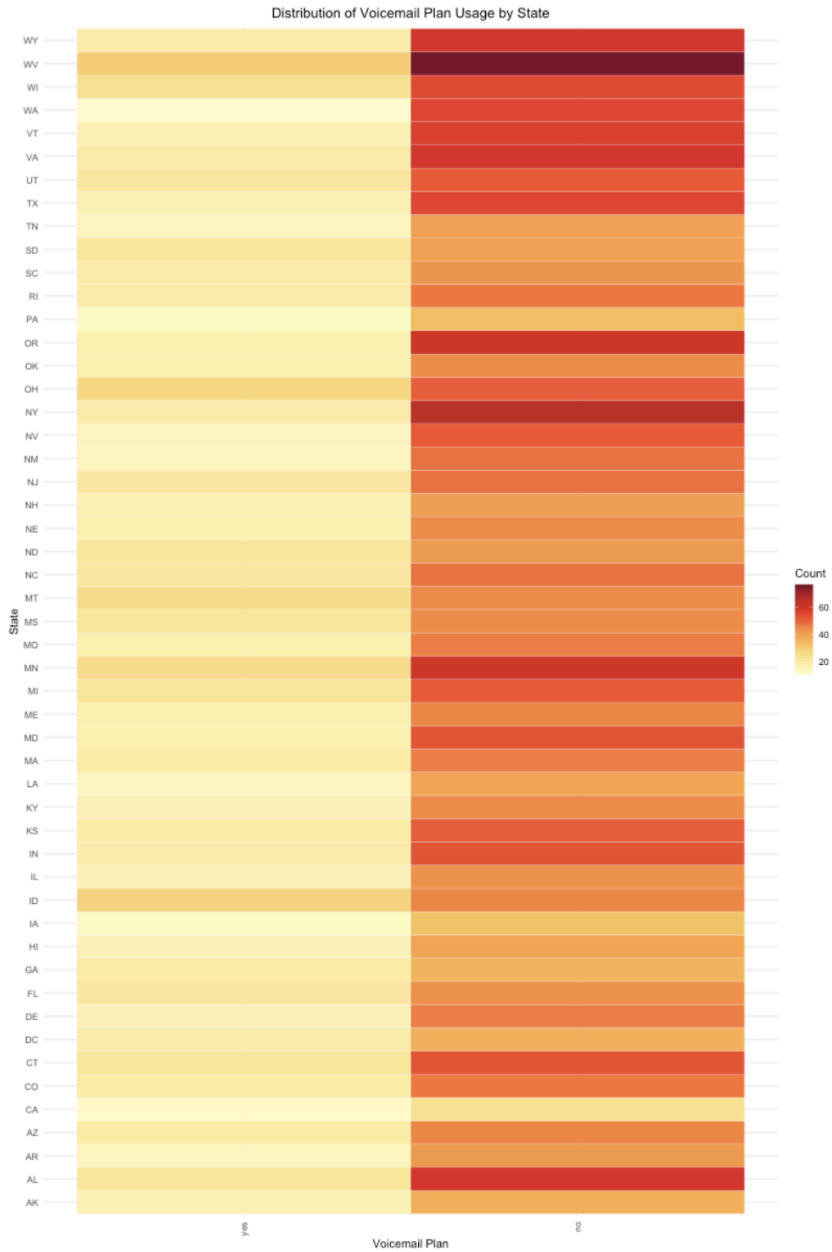


The histogram shows the distribution of customer service calls and the frequency rates. Between 1 and 2, the frequency of customer service calls increases to approximately 1180. But after that frequency, it starts decreasing and the lowest frequency is approximately 10 when the service call is 6.5.

4) Heatmap

Code:

```
state_vmail_plan <- churn_data %>%
group_by(State, VMail.Plan) %>%
summarise(count = n(), .groups = "drop") %>%
mutate(VMail.Plan = factor(VMail.Plan, levels = c("yes", "no"))) # Ensure proper
factor levels


colors <- colorRampPalette(brewer.pal(9, "YlOrRd"))(100) # Creates 100 shades from
Yellow to Orange to Red


plot <- ggplot(state_vmail_plan, aes(x = VMail.Plan, y = State, fill = count)) +
geom_tile(color = "white") + # White borders for distinction
scale_fill_gradientn(colors = colors) +
labs(
title = "Distribution of Voicemail Plan Usage by State",
x = "Voicemail Plan",
y = "State",
fill = "Count"
) +
theme_minimal() +
theme(
axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
plot.title = element_text(hjust = 0.5),
legend.position = "right"
)

print(plot)
```

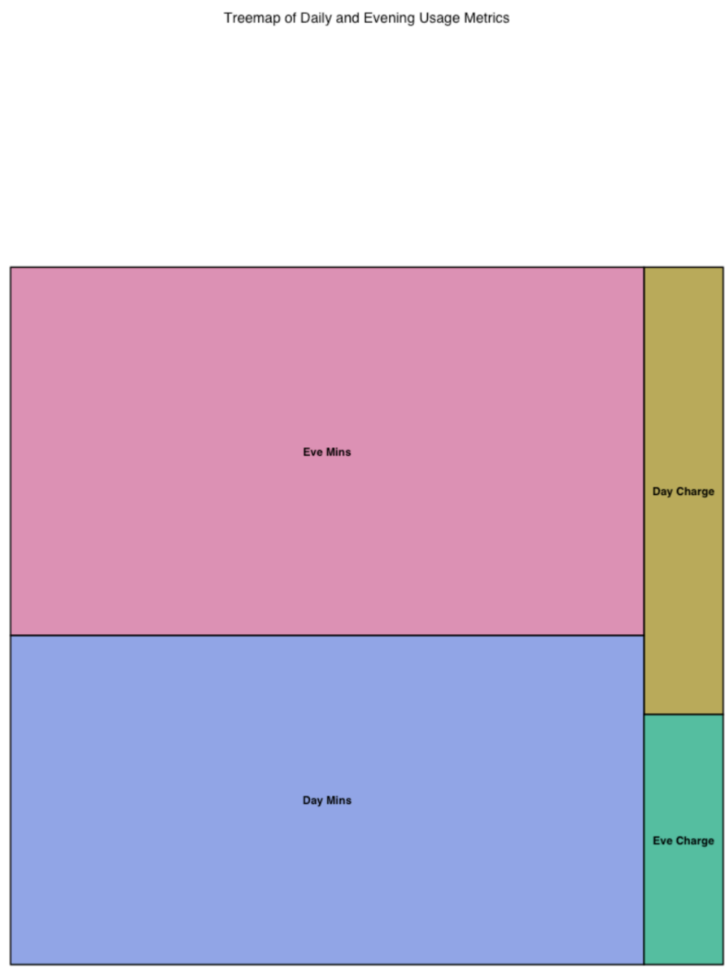


The heat map illustrates how many people use the voicemail plan in different states and how many people do not use it at all. Based on the color, there are some states that have a larger number of people using voicemail plans. Also there are some states that have a larger number of people who do not use voicemail plans. Some of these states have average numbers for both yes and no.

5) Treemap

Code:

```
state_vmail_plan <- churn_data %>%
group_by(State, VMail.Plan) %>%
summarise(count = n(), .groups = "drop") %>%
mutate(VMail.Plan = factor(VMail.Plan, levels = c("yes", "no")))
colors <- colorRampPalette(brewer.pal(9, "YlOrRd"))(100)
plot <- ggplot(state_vmail_plan, aes(x = VMail.Plan, y = State, fill = count)) +
geom_tile(color = "white") +
scale_fill_gradientn(colors = colors) +
labs(
title = "Distribution of Voicemail Plan Usage by State",
x = "Voicemail Plan",
y = "State",
fill = "Count"
) +
theme_minimal() +
theme(
axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
plot.title = element_text(hjust = 0.5),
legend.position = "right"
)

print(plot)
```



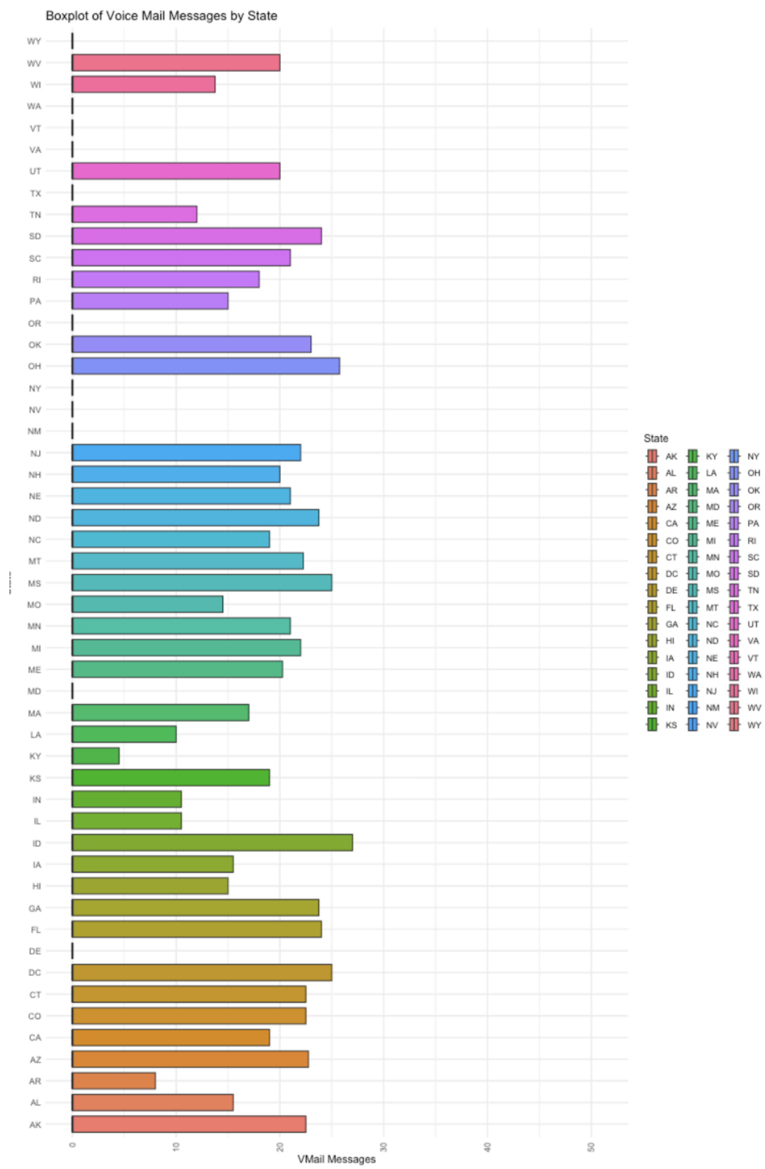
The treemap shows the day and evening calls usage metrics. It is pretty similar to the pie chart, but in the treemap, it is easier to look at the map and compare day minutes and charges with evening minutes and charges. Based on the treemap, the evening charge is the small part of this tree.

6) Boxplot

Code:

```
plot <- ggplot(data = churn_data, aes(x = VMail.Message, y = State, fill = State)) +
geom_boxplot(stat = "boxplot", coef = 0, outlier.shape = NA) +
labs(title = "Boxplot of Voice Mail Messages by State", x = "VMail Messages", y =
"State
") +
theme_minimal() +
```

```
theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))
print(plot)
```



The box plot illustrates the total number of voicemail messages based on states. According to the plot, the state of ID has the highest number of voicemail messages than other states. Using different colors for this plot would help us observe that each states the number of voicemail messages.

7) Doughnut Chart

Code:

```
counts <- c(
sum(churn_data$Day.Mins, na.rm = TRUE),
sum(churn_data$Day.Charge, na.rm = TRUE),
sum(churn_data$Night.Mins, na.rm = TRUE),
sum(churn_data$Night.Charge, na.rm = TRUE)
)

categories <- c("Day Mins", "Day Charge", "Night Mins", "Night Charge")
data <- data.frame(Category = categories, Value = counts)

plot <- ggplot(data, aes(x = "", y = Value, fill = Category)) +
geom_bar(stat = "identity", width = 1) +
coord_polar(theta = "y") +
theme_void() +
theme(legend.position = "bottom") +
scale_fill_manual(values = c("Day Mins" = "#006aff", "Day Charge" = "#ffa600", "Night
Mins" = "#fffb00", "Night Charge" = "#9500ff")) +
labs(title = "Distribution of Calls and Charges Day vs Night") +

# White circle
annotate("text", x = 0, y = 0, label = "", size = 10, color = "white")

print(plot)
```

This doughnut chart shows the distribution of calls and charges day vs night. The night charge is less than the day charge.

8) Naive Bayes

```
library(rpart)
library(rpart.plot)
data <- ("~/Desktop/CMPT 436/DataMiningProject_2024/churn.txt")
churn_data <- read.csv(data, stringsAsFactors = TRUE)
names(churn_data) <- c("State", "Account_Length", "Area_Code", "Phone_Number",
"International_Plan", "Voice_Mail_Plan", "Number_Of_Voice_Mail_Messages", "Total_Day_Minutes",
"Total_Day_Calls", "Total_Day_Charge", "Total_Eve_Minutes", "Total_Eve_Calls",
"Total_Eve_Charge", "Total_Night_Minutes", "Total_Night_Calls", "Total_Night_Charge",
"Total_International_Minutes", "Total_International_Calls", "Total_International_Charge",
"Number_Of_Calls_To_Customer_Service", "Churn")
df <- churn_data[-4]
churn_data$Churn <- factor(churn_data$Churn, levels = c("True", "False"))

set.seed(1234)
train <- sample(nrow(churn_data), 0.7 * nrow(df))
df.train <- df[train,]
df.validate <- df[-train, ]
table(df.train$Churn)
table(df.validate$Churn)

library(e1071)

nb.model <- naiveBayes(Churn~ ., data = df.train)
nb.pred <- predict(nb.model, df.validate)
nb.perf <- table(df.validate$Churn, nb.pred, dnn=c("Actual", "Predicted"))
nb.perf
tn <- nb.perf[1, 1]
fp <- nb.perf[1, 2]
fn <- nb.perf[2, 1]
tp <- nb.perf[2, 2]
```

```
accuracy <- (tp + tn) / (tp + tn + fp + fn)
accuracy
error_rate <- (fp + fn) / (tp + tn + fp + fn)
error_rate
sensitivity <- tp / (tp + fn)
sensitivity
specificity <- tn / (tn + fp)
specificity
precision <- tp / (tp + fp)
precision
f_measure <- (2 * precision * sensitivity) / (precision + sensitivity)
f_measure
```

```
--
False  True
 1984   349
> table(df.validate$Churn)

False  True
  866   134
>
> library(e1071)
>
> nb.model <- naiveBayes(Churn~ ., data = df.train)
> nb.pred <- predict(nb.model, df.validate)
>
> nb.perf <- table(df.validate$Churn, nb.pred, dnn=c("Actual", "Predicted"))
> nb.perf
       Predicted
Actual   False True
  False    820   46
  True      77   57
>
> tn <- nb.perf[1, 1]
> fp <- nb.perf[1, 2]
> fn <- nb.perf[2, 1]
> tp <- nb.perf[2, 2]
>
> # Calculate metrics
> accuracy <- (tp + tn) / (tp + tn + fp + fn)
> accuracy
[1] 0.877
> error_rate <- (fp + fn) / (tp + tn + fp + fn)
> error_rate
[1] 0.123
> sensitivity <- tp / (tp + fn)
> sensitivity
[1] 0.4253731
> specificity <- tn / (tn + fp)
> specificity
[1] 0.9468822
> precision <- tp / (tp + fp)
> precision
[1] 0.5533981
> f_measure <- (2 * precision * sensitivity) / (precision + sensitivity)
> f_measure
[1] 0.4810127
```

The accuracy for Naive Bayes is 87.7% which is less than 90%. So we cannot assume that Naive Bayes has the most accurate accuracy.

9) Decision Tree:

library(rpart)

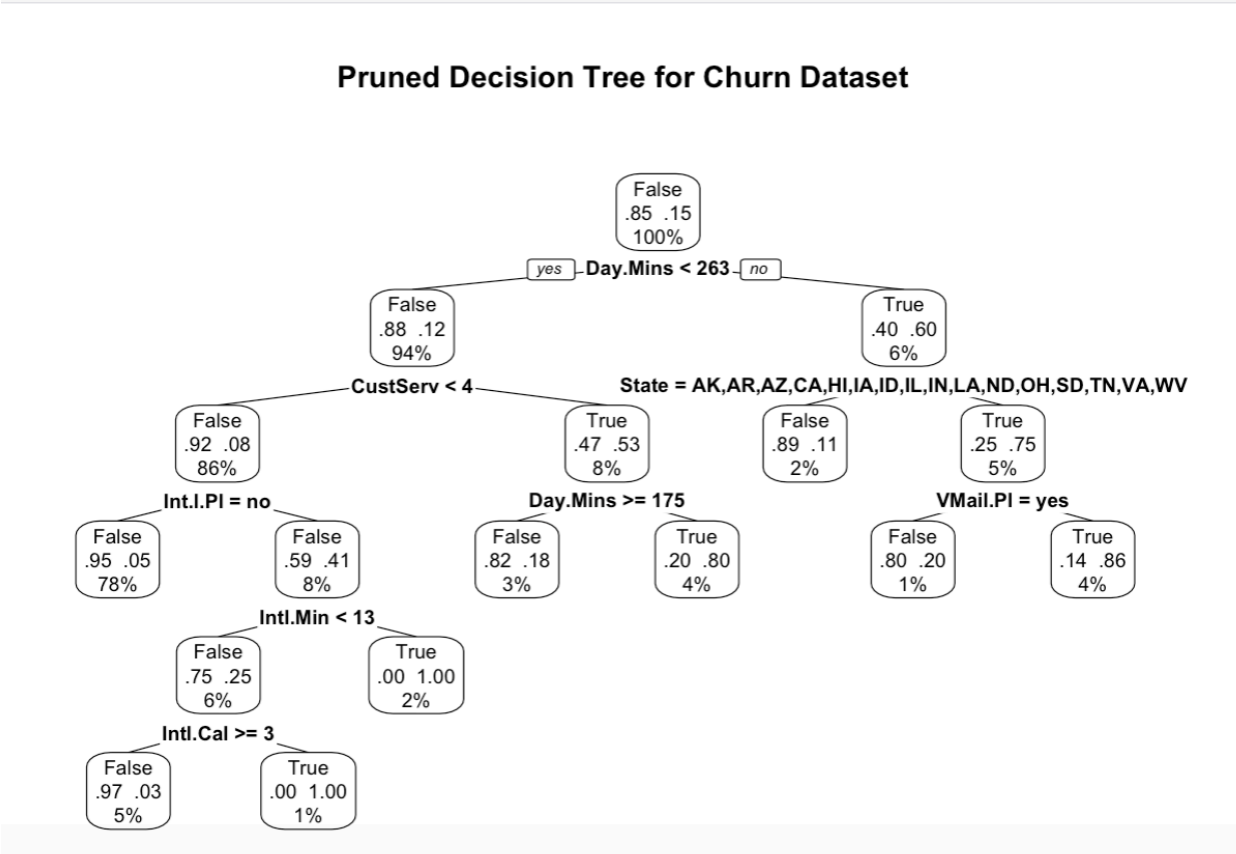
library(rpart.plot)

data <- ("~/Desktop/CMPT 436/churn.txt")

churn_data <- read.csv(data, stringsAsFactors = TRUE)

```
churn_data$Churn. <- factor(churn_data$Churn.)
df <- churn_data[, !(names(churn_data) %in% c("Phone"))]

# Set seed for reproducibility
set.seed(1234)
train_indices <- sample(nrow(df), 0.7 * nrow(df))
df.train <- df[train_indices, ]
df.validate <- df[-train_indices, ]

# Build a Decision Tree model
dtree <- rpart(Churn. ~ ., data=df.train, method="class", parms=list(split="information"))

# Prune the tree based on complexity parameter (this value would need tuning)
dtree_pruned <- prune(dtree, cp=dtree$cptable[which.min(dtree$cptable[,"xerror"]), "CP"])

# Plot the pruned decision tree
prp(dtree_pruned, type=2, extra=104, main="Pruned Decision Tree for Churn Dataset")

# Predict on validation set
dtree_pred <- predict(dtree_pruned, newdata=df.validate, type="class")

dtree_perf <- table(actual=df.validate$Churn., predicted=dtree_pred, dnn=c("Actual", "Predicted"))
# Extract true negatives, false positives, false negatives, and true positives
tn <- dtree_perf[1,1]
fp <- dtree_perf[1,2]
fn <- dtree_perf[2,1]
tp <- dtree_perf[2,2]

# Calculate metrics
accuracy <- (tp + tn) / (tp + tn + fp + fn)
error_rate <- (fp + fn) / (tp + tn + fp + fn)
sensitivity <- tp / (tp + fn)  # Also known as recall
specificity <- tn / (tn + fp)
precision <- tp / (tp + fp)
```

```
f_measure <- (2 * precision * sensitivity) / (precision + sensitivity)
```



```
> print(accuracy)
[1] 0.917
> print(error_rate)
[1] 0.083
> print(sensitivity)
[1] 0.5298507
> print(specificity)
[1] 0.9769053
> print(precision)
[1] 0.7802198
> print(f_measure)
[1] 0.6311111
>
```

```
> dtree$cptable
          CP nsplit rel error    xerror       xstd
1 0.08500478      0 1.0000000 1.0000000 0.04936291
2 0.08022923      3 0.7449857 0.9197708 0.04767423
3 0.05730659      4 0.6647564 0.8080229 0.04511538
4 0.03438395      7 0.4555874 0.5558739 0.03821408
5 0.01719198      8 0.4212034 0.5300860 0.03739557
6 0.01528176     10 0.3868195 0.5587393 0.03830349
7 0.01432665     13 0.3409742 0.5501433 0.03803436
8 0.01146132     15 0.3123209 0.5386819 0.03767123
9 0.01000000     17 0.2893983 0.5444126 0.03785342
```

The accuracy of Decision Tree is 91.7% which is better than the accuracy of Naive Bayes.

10) Support Vector Machine Classifier

library(rpart)

library(rpart.plot)

data <- ("~/Desktop/CMPT 436/churn.txt")

churn_data <- read.csv(data, stringsAsFactors = TRUE)

```
names(churn_data) <- c("State", "Account_Length", "Area_Code", "Phone_Number",
"International_Plan", "Voice_Mail_Plan", "Number_Of_Voice_Mail_Messages", "Total_Day_Minutes",
"Total_Day_Calls", "Total_Day_Charge", "Total_Eve_Minutes", "Total_Eve_Calls",
"Total_Eve_Charge", "Total_Night_Minutes", "Total_Night_Calls", "Total_Night_Charge",
"Total_International_Minutes", "Total_International_Calls", "Total_International_Charge",
"Number_Of_Calls_To_Customer_Service", "Churn")

# remove 'Phone_Number' column
df <- churn_data[-4]

# Convert target variable 'Churn' to a factor
df$Churn <- factor(df$Churn, levels = c("False", "True"))

set.seed(1234)
train <- sample(nrow(df), 0.7 * nrow(df))
df.train <- df[train, ]
df.validate <- df[-train, ]
table(df.train$Churn)
table(df.validate$Churn)

library(e1071)

svm.model <- svm(Churn~., data=df.train)
svm.pred <- predict(svm.model, na.omit(df.validate))
svm.perf <- table(na.omit(df.validate)$Churn, svm.pred, dnn=c("Actual", "Predicted"))
svm.perf

tn <- svm.perf[1,1]
fp <- svm.perf[1,2]
fn <- svm.perf[2,1]
tp <- svm.perf[2,2]

# Calculate metrics
accuracy <- (tp + tn) / (tp + tn + fp + fn)
```

accuracy

error_rate <- (fp + fn) / (tp + tn + fp + fn)

error_rate

sensitivity <- tp / (tp + fn) # Also known as recall

sensitivity

specificity <- tn / (tn + fp)

specificity

precision <- tp / (tp + fp)

precision

f_measure <- (2 * precision * sensitivity) / (precision + sensitivity)

f_measure

```
> svm.model <- svm(Churn~., data=df.train)
> svm.pred <- predict(svm.model, na.omit(df.validate))
> svm.perf <- table(na.omit(df.validate)$Churn, svm.pred, dnn=c("Actual", "Predicted"))
> svm.perf
       Predicted
Actual  False True
  False   864    2
  True    106   28
```

```
> accuracy <- (tp + tn) / (tp + tn + fp + fn)
> accuracy
[1] 0.892
> error_rate <- (fp + fn) / (tp + tn + fp + fn)
> error_rate
[1] 0.108
> sensitivity <- tp / (tp + fn)  # Also known as recall
> sensitivity
[1] 0.2089552
> specificity <- tn / (tn + fp)
> specificity
[1] 0.9976905
> precision <- tp / (tp + fp)
> precision
[1] 0.9333333
> f_measure <- (2 * precision * sensitivity) / (precision + sensitivity)
> f_measure
[1] 0.3414634
```

The accuracy of the support vector machine classifier is 89.2% which is less than the accuracy of Decision Tree but better than the accuracy of Naive Bayes. Because of different types of classifications, it is certain that the accuracy will be different. For the Churn data, the Decision Tree is more accurate because of the accuracy.